

Pizza Trader (obsolete)

An exercise in Fusion Tables INSERT, UPDATE, SELECT for App Inventor 2

9/2017

[Background](#)

[Prerequisites](#)

[Pizza Trades Table](#)

[Designer](#)

[Blocks](#)

[Login](#)

[Screen1.Initialize](#)

[when _ntfLogin_AfterTextInput](#)

[Screen shot - just signed in](#)

[Presenting Available Pizza](#)

[requestAvailable](#)

[When ftbAvailable.GotResult](#)

[lpkAvailable](#)

[When lpkAvailable.AfterPicking](#)

[confirmAcceptance](#)

[Acceptance Notification sample](#)

[When ntfAccept.AfterChoosing](#)

[Accepting a Pizza Slice](#)

[Update](#)

[when ftbUpdate.GotResult](#)

[Offering New Pizza](#)

[When btnOffer.Click](#)

[when btnCancel_Click](#)

[when lpkTypes_AfterPicking](#)

[Pizza type entry](#)

[when btnPost_Click](#)

[when ftbINSERT_GotResult](#)

[Available Pizza Display](#)

[Data Flow](#)

[Appendix](#)

[Creating a Service Account](#)

[Get a .p12 file](#)

[Copy the service account email address to the clipboard](#)

[Pull in a Fusion Tables control](#)

[Sharing the Table with the Service Account Email](#)

[Copy the table_ID for the Blocks Editor](#)

[table_ID](#)

[This app in the Gallery](#)

[More Projects](#)

Background

This is a simple single table Fusion Tables app for an imaginary fraternity house $\iota\eta\pi$ (Iota Eta Pi) to reduce food waste by sharing excess pizza amongst themselves. It should allow:

- Over stuffed frat house members to offer up their extra slices of pizza to others
- Hungry frat members to see what extra slices of pizza are on offer and by whom
- Pizza recipients to mark the offered slices as accepted by them
- Members to see who is being generous and who is being a freeloader
- College health department to track food poisoning outbreaks back to their source

Warning; Google has shut down their Fusion Tables service. I leave this tutorial up so people can study its design.

Prerequisites

This tutorial is intended for people who have completed the starter tutorial Pizza Party successfully at <http://appinventor.mit.edu/explore/ai2/pizzaparty.html> and know how to set up a service account for a Fusion Table.

Pizza Trades Table

Pizza Trades

For the AI2 Pizza Trader sample app

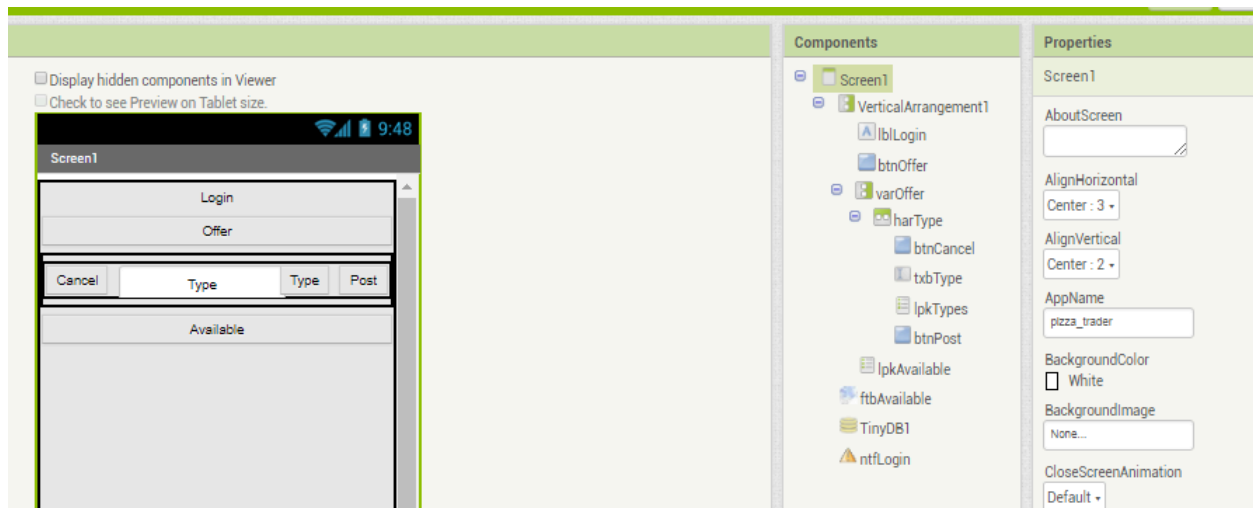
[Add Attribution](#) - Edited on 2017 September 5

File	Edit	Tools	Help	≡ Rows 1 ▾	☰ Cards 1	☰ Pizza li
Filter ▾		No filters applied				
⏮ ⏪ 1-5 of 5 ⏩ ⏭						
Type	FromName	FromDate	ToName	ToDate		
pineapple	ABG	2017-09-05	Don Ho	2017-09-05		
anchovies	Don Ho	2017-09-04	Sun Lee	2017-09-06		
Sicilian	Sun Lee	2017-09-05	ABG	2017-09-05		
pepperoni	ABG	2017-09-04	-			
pineapple	Sun Lee	2017-09-05	-			

Five slices of pizza were offered, but only three were accepted (To Name not equal to '-', the default value.)

For details on how to set up this table and its service account and one of its Fusion Tables controls, see [Creating a Service Account](#) in the Appendix.

Designer



From the top down, Screen1 has:

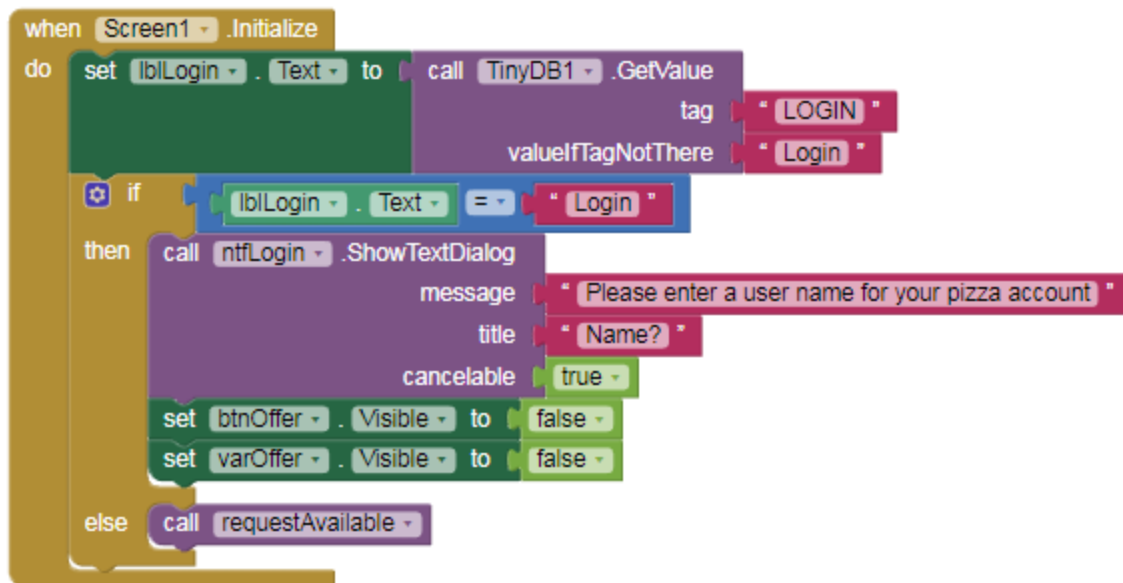
- A Login label. If logged in, the name from TinyDB is shown, else "Login".
- An Offer button, used to display the vertical arrangement varOffer
 - A Cancel button, to back out and hide varOffer
 - A text box to type in a new kind of pizza or display the selection from lpkTypes
 - A List Picker to pop up an easy list of pizza types to offer
 - A Post button to set up and launch the INSERT into the Pizza Trades table, if the User is logged in.
- An Available List Picker, to show available (ToName = "-") slices.

Non-visible components include TinyDB (for Login retention), some Fusion Table controls for INSERT, SELECT, UPDATE, and some Notifiers for various situations.

Blocks

Login

Screen1.Initialize

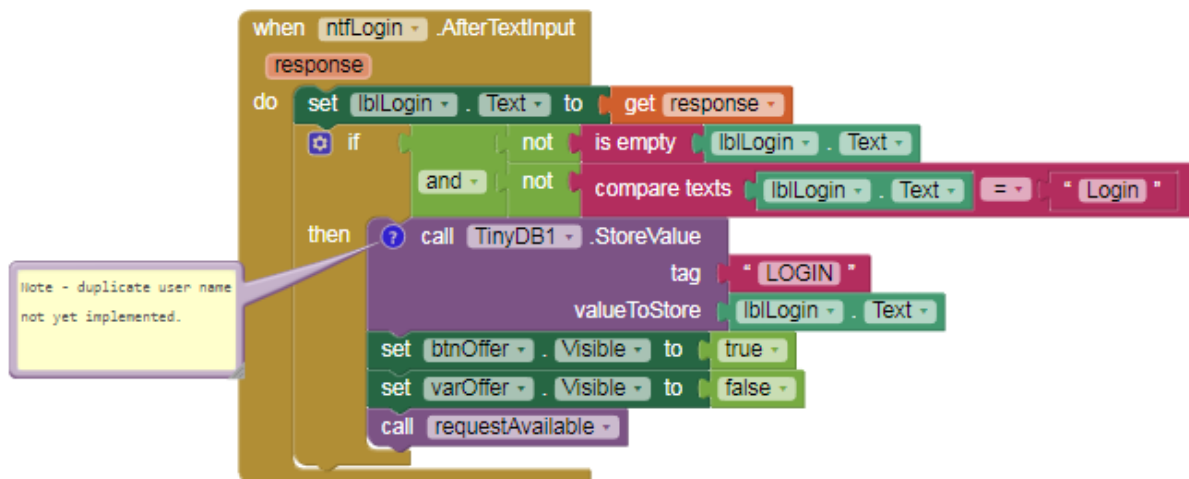


The user name is stored in TinyDB under tag 'LOGIN'.

You must supply a name to offer or withdraw pizza.

The user gets only one chance to enter a name, via a Notifier.

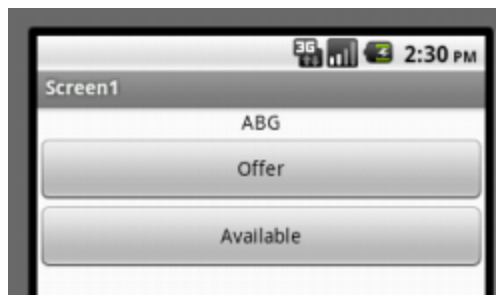
when_ntfLogin_AfterTextInput



This is a very loose registration process, since this is just for a frat house where every one knows one another. No checking is made for names already used to donate or receive pizza.

Once the name is available, the Offer button is enabled, and procedure [requestAvailable](#) is called to ask Fusion Tables for available pizza for the List Picker.

Screen shot - just signed in



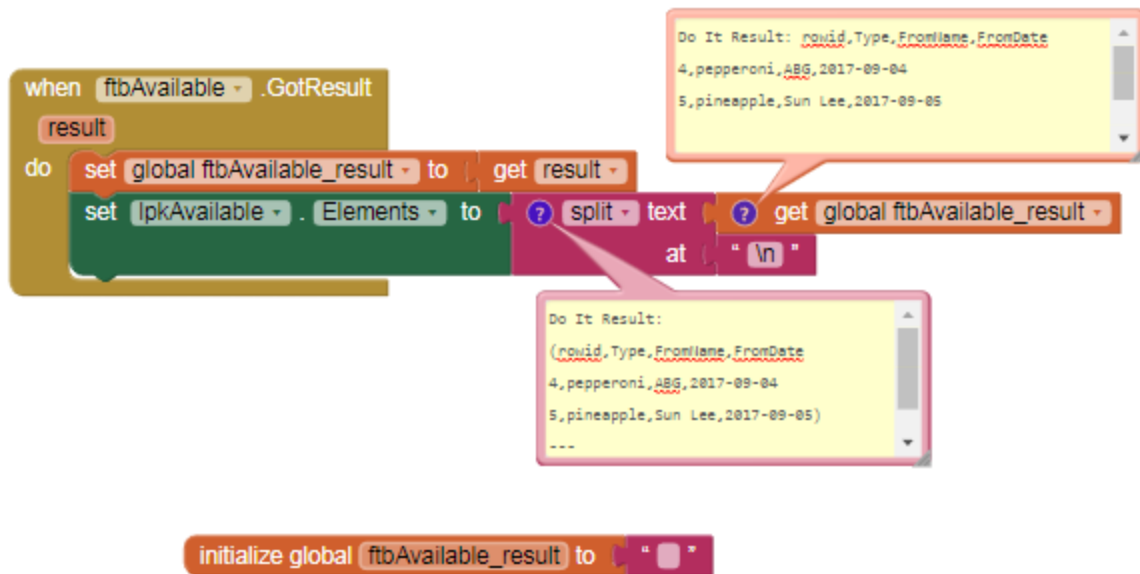
Presenting Available Pizza

`requestAvailable`



This procedure sets up the query text for the Fusion Table control `ftbAvailable`, then sends the query. Notice how we ask for ROWID, which is not visible in [Pizza Trades Table](#). We will need that later, to get to records in the Fusion Table. Also, notice the WHERE clause, where we ask for only the pizza that has not yet been given away. After sending the query, there is nothing more that can be done, until the Fusion Table control's `.GotResult` triggers.

When `ftbAvailable.GotResult`



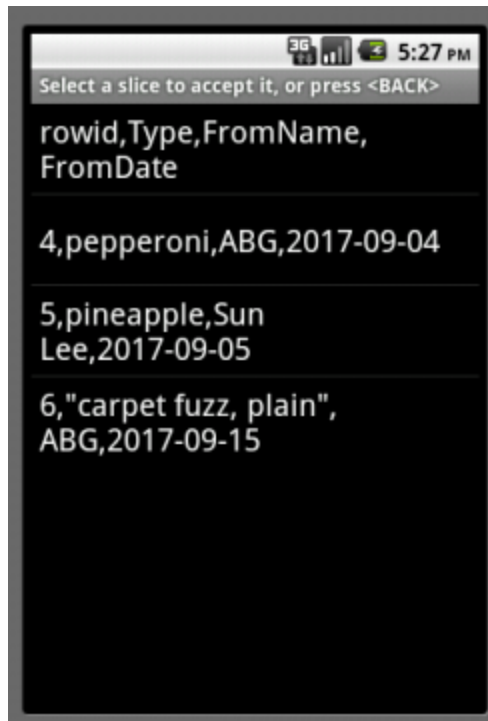
I'm doing some things differently from other tutorials, in how I accept data from Fusion Tables, for a reason ...

I have devoted a global variable to a copy of the result value, for debugging purposes. I would not have been able to apply a Do It to the result as I did, if it were a transient local variable.

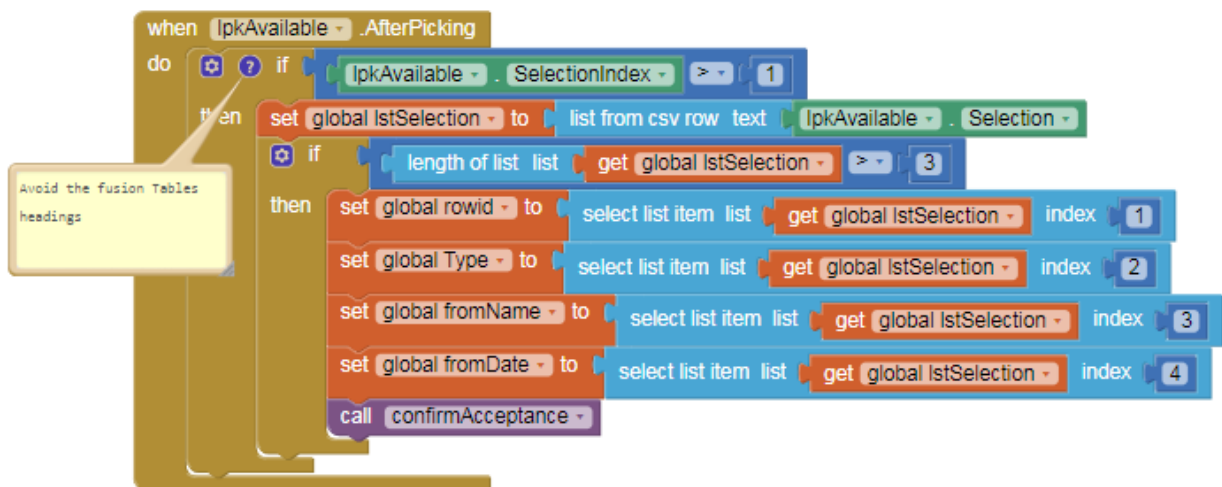
I also have avoided use of the **list from csv table** conversion block, instead just splitting the incoming text result at the line feed (`\n`) characters, into a one dimensional list of comma separated text rows, ready for use as an Elements list in my List Picker `lpkAvailable`. I want those commas in there, to delimit my fields and to avoid having to strip off road kill `()` wrappers left over from list to text auto conversion.

No table was used here, just text and a list of text.

`lpkAvailable`

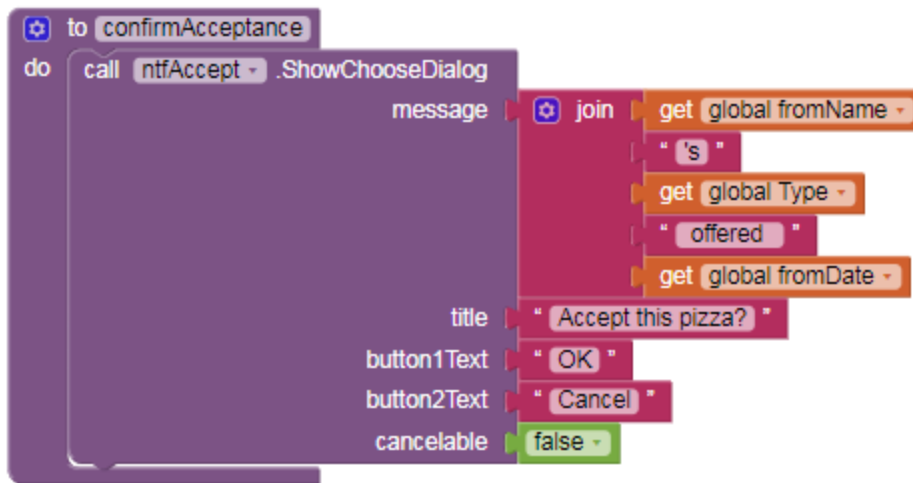


When lpkAvailable.AfterPicking



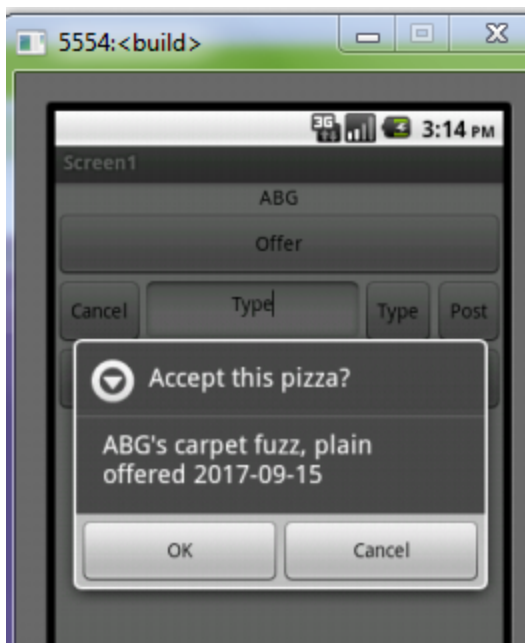
The List Picker lpkAvailable has Elements for all the currently available pizza. Each Element has a csv text containing rowid, Type, fromName, and fromDate for that slice of pizza. If we select a slice, we call routine [confirmAcceptance](#) to verify the selection.

confirmAcceptance



Note that we do not complain if a person wants to accept pizza that he donated. This would be a good place to mention that, and delete the row if he donated it instead of updating it with his acceptance.

Acceptance Notification sample



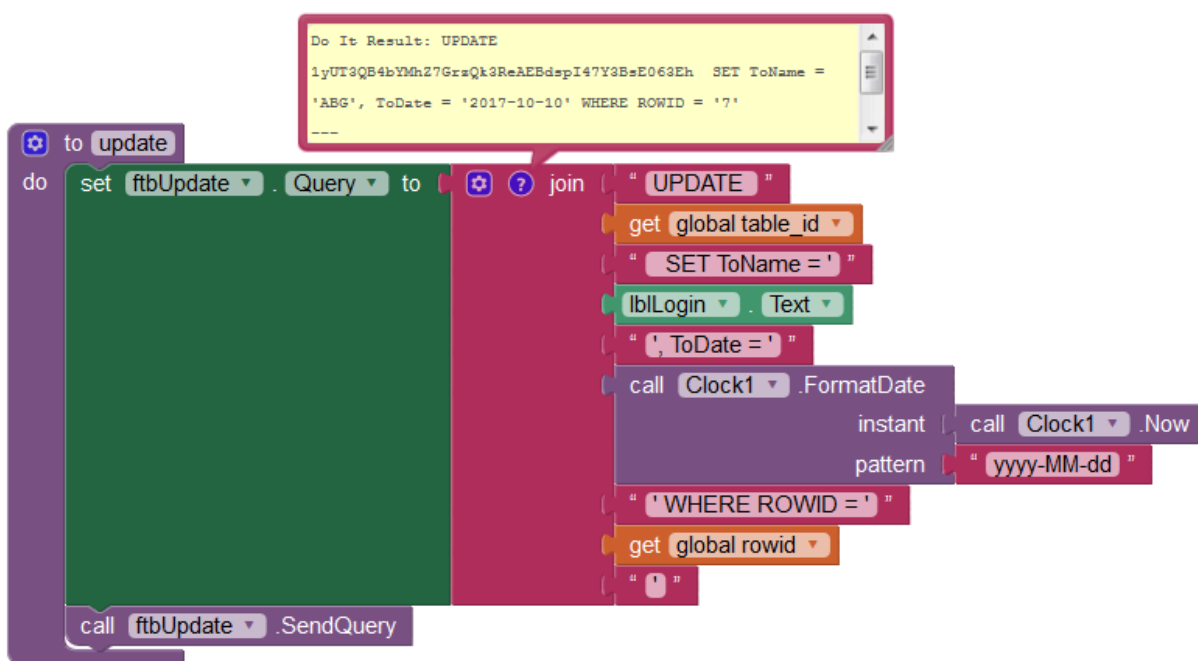
When *ntfAccept.AfterChoosing*



If the user okays the update, call procedure [update](#)

Accepting a Pizza Slice

Update



The **rowid** global was set in [When lpkAvailable.AfterPicking](#) after the user selected from [lpkAvailable](#).¹

This is the query syntax for an update in Fusion Tables:

¹ Astute readers will notice that my rowid value of '7' is missing from my illustration of the List Picker showing available pizza. That's because I took the screen shots in the wrong order, after slice 7 became unavailable, and am too lazy to do them over again in the right order - ABG.

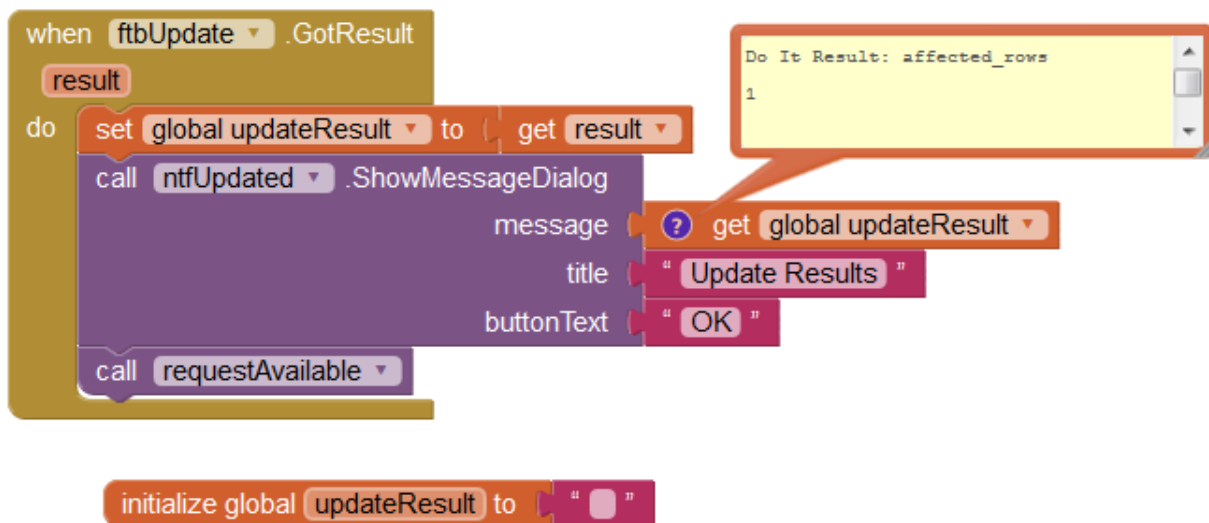
To [update](#) values in one or more columns of an existing row in a table, use the following syntax in an HTTP POST request:

```
UPDATE <table_id>  
SET <column_name> = <value> {, <column_name> = <value> }*  
WHERE ROWID = <row_id>
```

Returns

If the request succeeds, the server responds with a 200 OK status code and the number of affected rows in [JSON format](#).

when ftbUpdate.GotResult



We are taking the lazy way out after the Update, displaying the result message without looking at it, leaving the interpretation for the app user. Notice the `\n` between `"affected_rows"` and the `"1"` in the result.

Since the available slices have changed in the table, we need to call [requestAvailable](#) to refresh the list picker from the Fusion table.

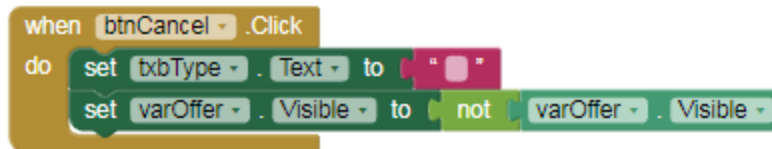
Offering New Pizza

When btnOffer.Click



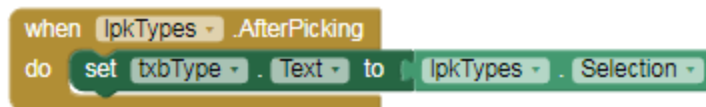
This button toggles the visibility of the input panel for offering pizza.

when btnCancel.Click



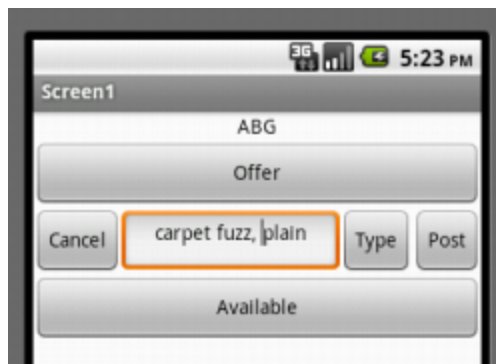
This button lets you cancel submission of a pizza offer. It just needs to hide its enclosing arrangement, and clear the type.

when lpkTypes.AfterPicking



For convenience, common pizza types are preloaded in the List Picker to help fill the text box pizza type. If you want to get fancy, build up complex types using text JOIN blocks.

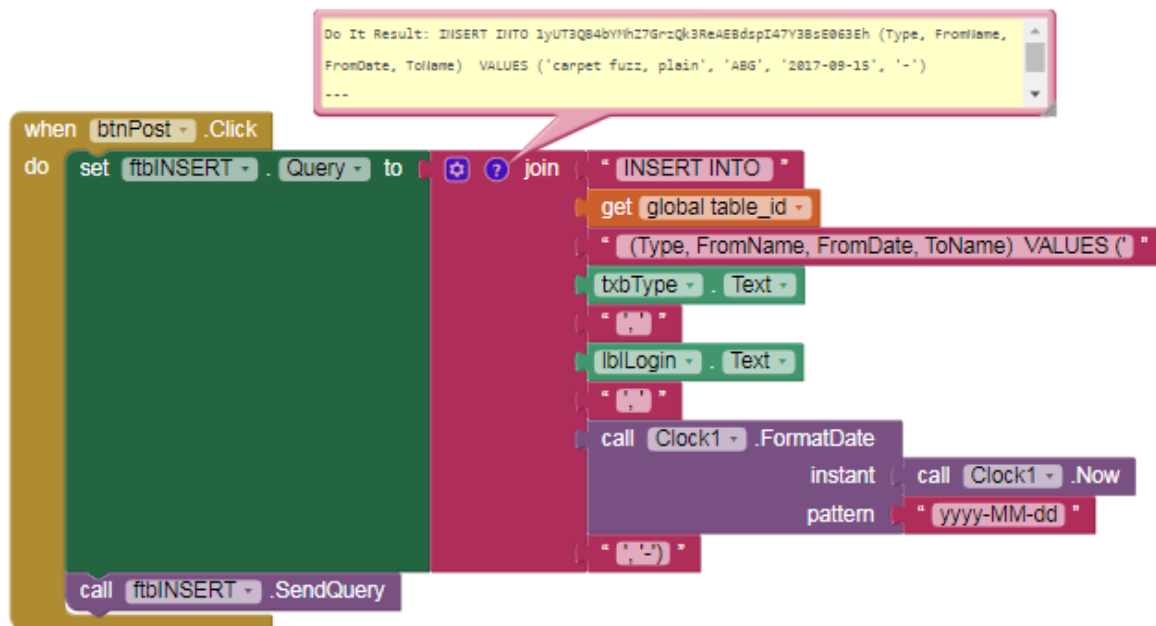
Pizza type entry



In this case, I have combined the stock type **plain** with some more unfortunate details about my slippery pizza slice. I next hit the **Post** button.

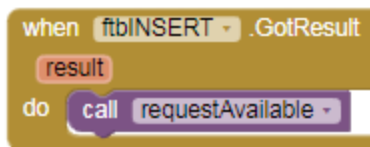
when btnPost_Click

This is where we send an INSERT to Fusion Tables for our pizza offer. The INSERT syntax is at <https://developers.google.com/fusiontables/docs/v2/sql-reference#insertRow>.



When building a query, be careful to double check that your spaces after the word INTO have not been eaten by the Blocks Editor. I usually go back and retype the end of the text block, to add the lost trailing space. This is a common cause of syntax errors in SQL. Also notice the balanced parentheses around names and values, and the apostrophes (') around text values. I chose yyyy-mm-dd as my date format to get the benefit of chronological order matching text sort order.

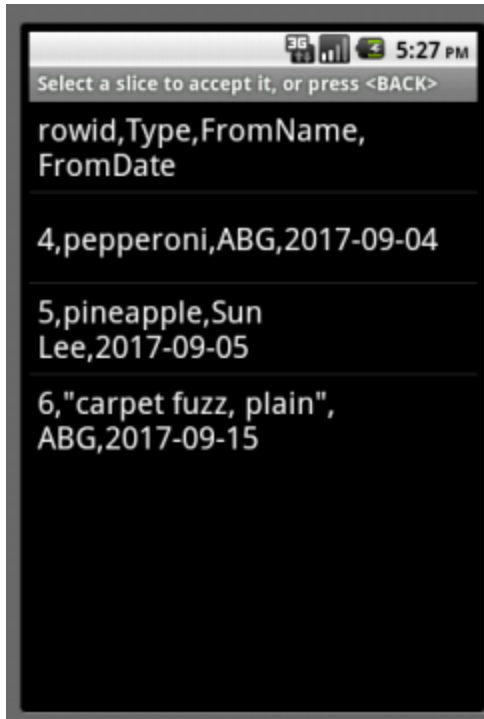
when ftbINSERT_GotResult



If the INSERT succeeded, we will have to refresh the available pizza List Picker Elements using procedure [requestAvailable](#).

We can double check that it arrived by hitting the Available List Picker:

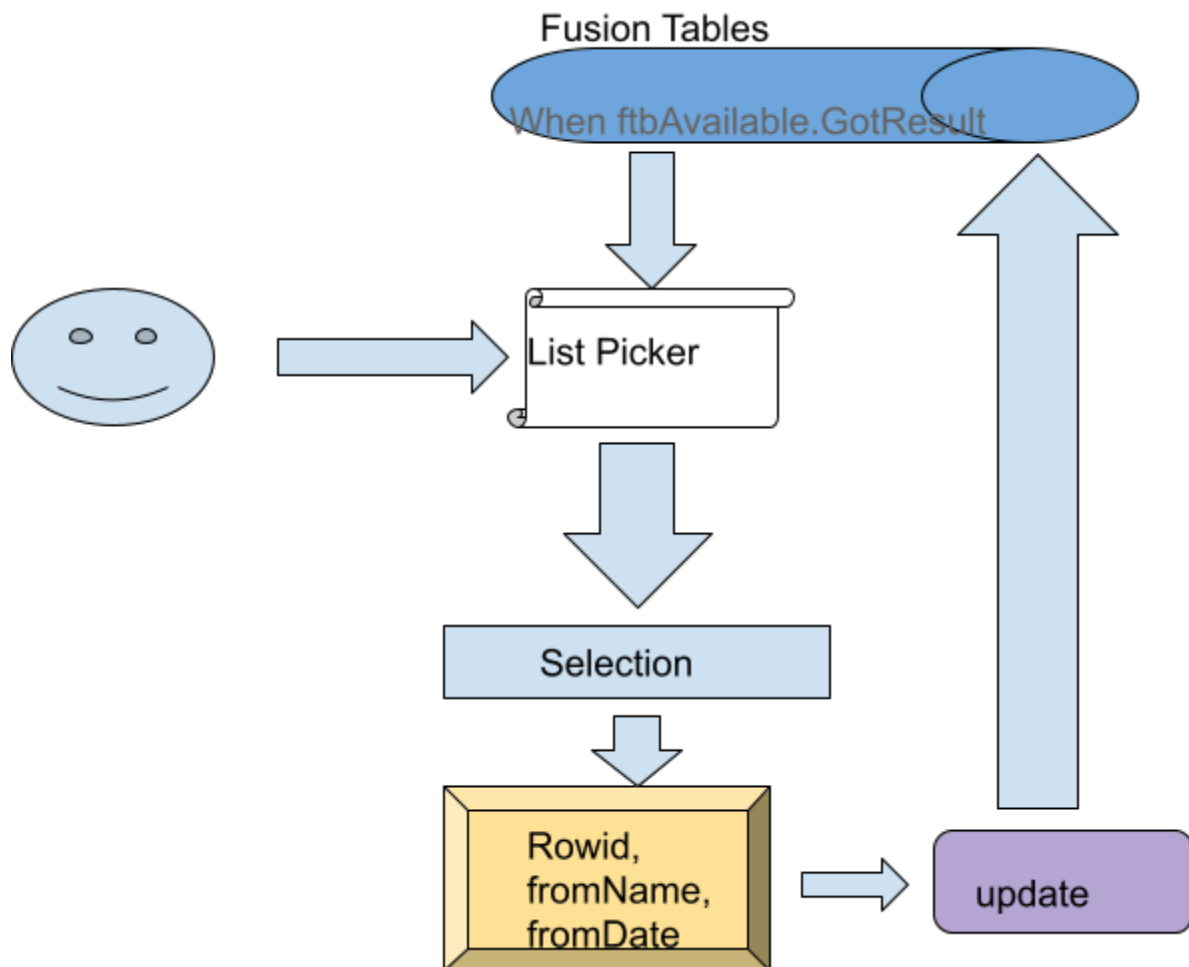
Available Pizza Display



rowid	Type	FromName	FromDate
4	pepperoni	ABG	2017-09-04
5	pineapple	Sun Lee	2017-09-05
6	"carpet fuzz, plain"	ABG	2017-09-15

Notice how the comma in the pizza type is protected by double quotes, so it doesn't get confused with the commas separating columns in the table.

Data Flow



Appendix

Creating a Service Account

Go to the Google Developers Console at <https://console.developers.google.com/>

Like our APIs? Check out our infrastructure. Sign up to get \$300 in credit and 12 months to explore Google Cloud Platform. [Learn more](#) DISMISS SIGN UP FOR FREE TRIAL

Google APIs API Project

APIs & services

- Dashboard
- Library
- Credentials

Dashboard

[+ ENABLE APIS AND SERVICES](#)

Enabled APIs and services

Some APIs and services are enabled automatically

Activity for the last hour

1 hour 6 hours 12 hours 1 day 2 days 4 days 7 days 14 days 30 days

Traffic

Requests/sec

There is no traffic for this time period.

Errors

Percent of requests

There are no errors for this time period.

Median latency

Milliseconds

There is no latency data.

API	Requests	Errors	Error ratio	Latency, median	Latency, 98%
Fusion Tables API	—	—	—	—	—
					Disable

Select **credentials**. Do **not** sign up for the free trial.

Like our APIs? Check out our infrastructure. Sign up to get \$300 in credit and 12 months to explore Google Cloud Platform. [Learn more](#) DISMISS SIGN UP FOR FREE TRIAL

Google APIs API Project

APIs & services

- Dashboard
- Library
- Credentials

Credentials

[Credentials](#) [OAuth consent screen](#) [Domain verification](#)

[Create credentials](#) [Delete](#)

Create credentials to access your enabled APIs. [Refer to the API documentation](#) for details.

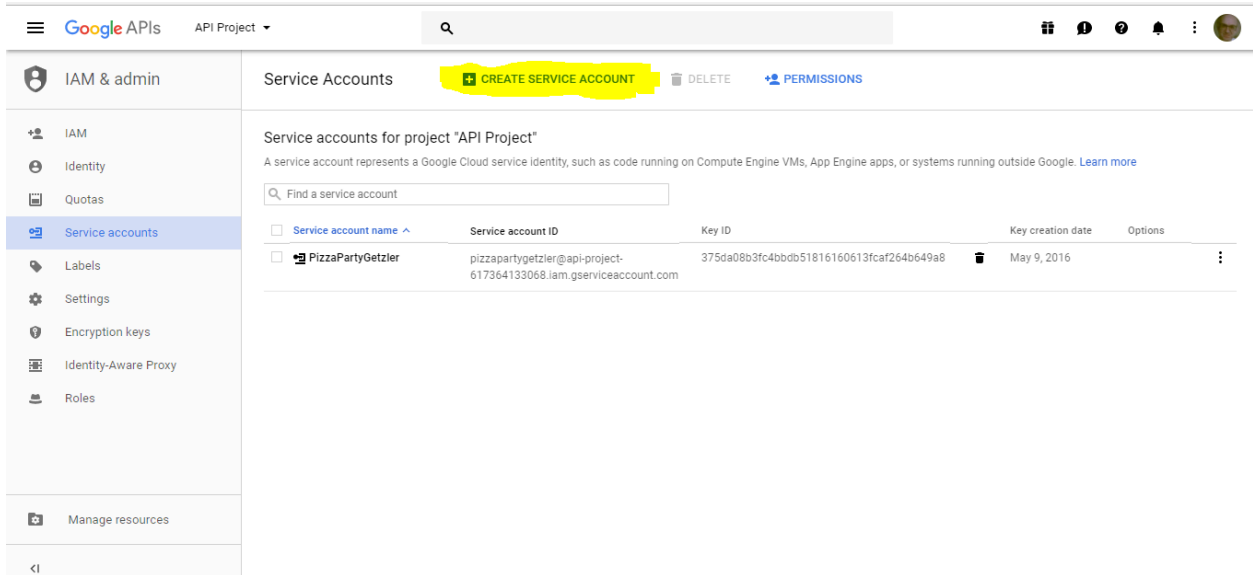
API keys

<input type="checkbox"/>	Name	Creation date	Restriction	Key	
<input type="checkbox"/>	Android key 1	Jan 29, 2014	None	AlzaSy8YrBcgU1X_gZb4PbeVKz8-i3BR11AuvglU	Manage service accounts

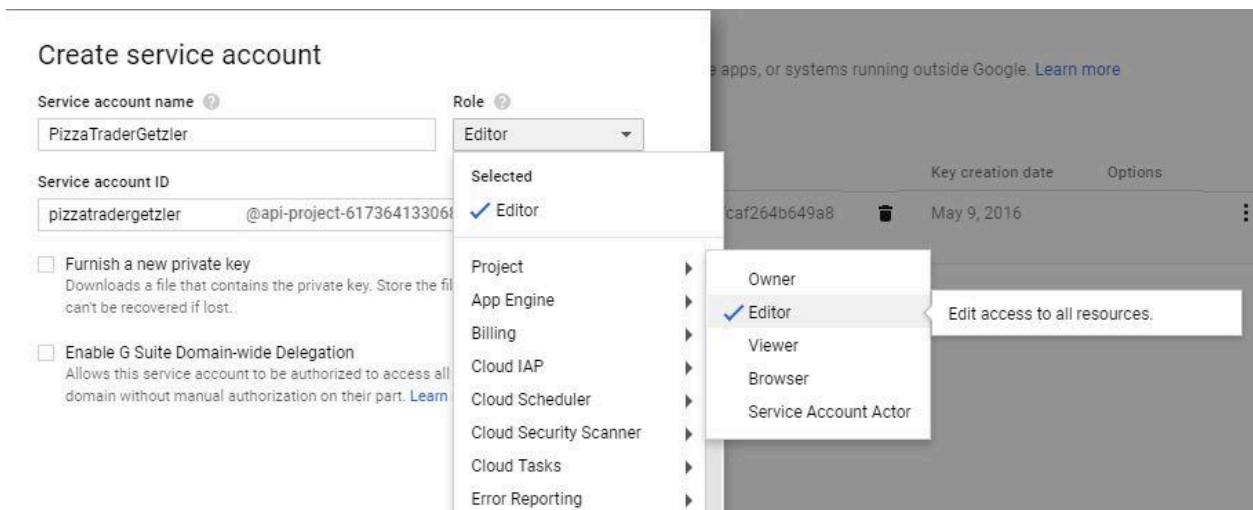
Service account keys

<input type="checkbox"/>	ID	Creation date	Service account	
<input type="checkbox"/>	375da08b3fc4bbdb51816160613fcdf264b649a8	May 9, 2016	PizzaPartyGetzler	

At the lower right of the Credentials section, under **Service account keys**, follow the link **Manage service accounts**.



We are now in the **Service Accounts** branch of **IAM & admin**. Follow the **CREATE SERVICE ACCOUNT** link.



We enter the new account name **PizzaTraderGetzler** and for good measure give it a role of Project->Editor.

Get a .p12 file

Create service account

Service account name ?	Role ?
<input type="text" value="PizzaTraderGetzler"/>	<input type="text" value="Editor"/>
Service account ID	
<input type="text" value="pizzatradergetzler @api-project-617364133068.iam.gserviceacco"/>	

You don't have permission to furnish a new private key.

- ☒ **Furnish a new private key**
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

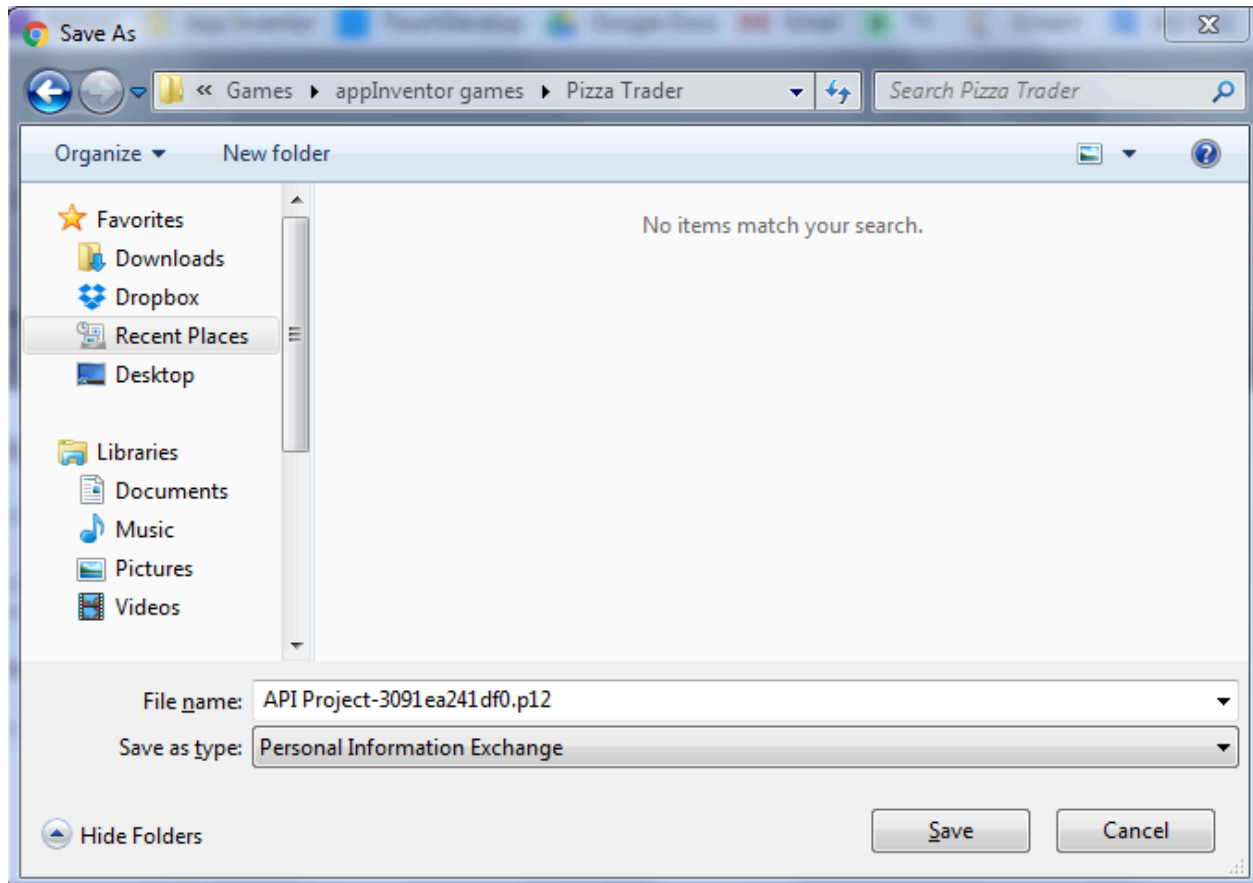
- ☐ JSON
Recommended
- ☒ P12
For backward compatibility with code using the P12 format

You don't have permission to modify the domain-wide delegation setting You don't have permission to modify the product name for the consent screen

- ☐ **Enable G Suite Domain-wide Delegation**
Allows this service account to be authorized to access all users' data on a G Suite domain without manual authorization on their part. [Learn more](#)

[CANCEL](#) [CREATE](#)

We then ask for a .P12 private key for compatibility with AI2's Fusion Table component.



Save the .p12 file, for uploading into the Media folder in AI2.

Service account and key created

New service account **PizzaTraderGetzler** has been created.

The account's private key **API Project-3091ea241df0.p12** has been saved on your computer. This is the only copy of the key, so store it securely.

This is the private key's password. It will not be shown again. You must present this password to use the private key. [Learn more](#)

notasecret

[CLOSE](#)

This is just a confirmation that the creation is complete, and a not very secret secret.

Copy the service account email address to the clipboard

The screenshot shows the Google Cloud IAM & Admin console. The left sidebar has a menu with 'IAM & admin' selected, and sub-items: IAM, Identity, Quotas, Service accounts (highlighted), Labels, Settings, and Encryption keys. The main area is titled 'Service Accounts' with buttons for 'CREATE SERVICE ACCOUNT', 'DELETE', and 'PERMISSIONS'. Below this, it says 'Service accounts for project "API Project"'. A search bar is present. A table lists two service accounts:

<input type="checkbox"/>	Service account name ^	Service account ID	Key ID	Key creation date
<input type="checkbox"/>	PizzaPartyGetzler	pizzapartygetzler@api-project-617364133068.iam.gserviceaccount.com	375da08b3fc4bbdb51816160613caf264b649a8	May 9, 2016
<input type="checkbox"/>	PizzaTraderGetzler	pizzatradergetzler@api-project-617364133068.iam.gserviceaccount.com	3091ea241df0f60a79bdfdd214320ff73d063f10	Sep 11, 2017

Pull in a Fusion Tables control

The screenshot shows the Fusion Tables control configuration in an Android Studio interface. The left pane shows a hierarchy of components: Screen1, VerticalArrangement1, btnLogin, btnOffer, varOffer, harType, txbType, lpkTypes, lpkAvailable, ftbAvailable (highlighted), and TinyDB1. The right pane shows the properties for the 'ftbAvailable' control:

- ftbAvailable
- ApiKey: [empty text box]
- KeyFile: APIProject-3091ea241df0...
- Query: show tables
- ServiceAccountEmail: pizzatradergetzler@api-pro...
- UseServiceAuthentication: [checked checkbox]

This Fusion Tables control is meant for use with a query to show all available pizza slices, hence the name I chose. I have pasted in

1. the .p12 file name, selected from the Media Drawer where I uploaded it,
2. The service account email, copied from the API Project page, and
3. Checked the Use Service Authentication checkbox.
4. I also left the ApiKey box empty.

Sharing the Table with the Service Account Email

Pizza Trades

For the AI2 Pizza Trader sample app
[Add Attribution](#) - Edited on 2017 September 11

File Edit Tools Help Rows 1 Cards 1

Filter No filters applied





1-5 of 5

Type	FromName	FromDate	ToName	ToDate
pineapple	ABG	2017-09-05	Don Ho	2017-09-05
anchovies	Don Ho	2017-09-04	Sun Lee	2017-09-04
Sicilian	Sun Lee	2017-09-05	ABG	2017-09-05
pepperoni	ABG	2017-09-04	-	-
pineapple	Sun Lee	2017-09-05	-	-



Sharing settings

Link to share


<https://www.google.com/fusiontables/DataSource?docid=1yUT3QB4bYMHZ7GrzQk3F>

Share link via:    

Who has access

	Public on the web - Anyone on the Internet can find and view	Change...
	Abraham Getzler (you) agetzler@gmail.com	Is owner

Invite people:

 pizzatradergetzler@api-project-617364133068.iam.gserv... x

Add more people...

☐ Notify people - [Add message](#)

[OK](#) [Cancel](#)

From the Google Docs Pizza Trades table, I select the Sharing button at the top right corner, and fill in an invitation to my service account email address, with edit capability. For instructional purposes, I left the table public for any one to view. That wasn't necessary for the app to work.

Sharing settings






Link to share

<https://www.google.com/fusiontables/DataSource?docid=1yUT3QB4bYMhZ7GrzQk3F>

Share link via:



Who has access

	Public on the web - Anyone on the Internet can find and view	Change...
	Abraham Getzler (you) agetzler@gmail.com	Is owner
	pizzatradergetzler@api-project-617364133...	 

Copy the table_ID for the Blocks Editor

At the Sharing settings pop up, the Link to share is highlighted, ready to copy. Notice at the end, it has **docid=...** with a very long ID key, which you will need to copy and paste into the Blocks Editor, into a global variable text box for use as the AI2 table_ID.

table_ID

```
initialize global table_id to "1yUT3QB4bYMhZ7GrzQk3ReAEBdSpl47Y3BsE063Eh"
```

This app in the Gallery

ai2.appinventor.mit.edu/?galleryId=4775793106092032

More Projects

https://docs.google.com/document/d/1acg2M5KdunKjJgM3Rxy_Rf6vT6OozxdIWglgbmzroA/edit?usp=sharing

